Requirements for Supporting Dynamic and Adaptive Workflow on the WWW

Peter J. Kammer
Gregory Alan Bolcer
Mark Bergman
Information and Computer Science
University of California, Irvine
Irvine, CA 92697-3425
1.949.824.8438
pkammer@ics.uci.edu, gbolcer@endeavors.org, mbergman@ics.uci.edu

CSCW'98 Workshop on Adaptive Workflow Systems

ABSTRACT

The unpredictability of business processes require that workflow systems support exception handling with the ability to dynamically adapt to the changing environment. Traditional approaches to handling this problem have fallen short, providing little support for change, particularly once the process has begun execution. We discuss some of the drawbacks of the traditional approaches, then identify and explain a number of key elements for supporting adaptive workflow: contingency management and hand-off, partial execution, dynamic behaviors, reflexivity, and process evolution and optimization over time. Together, these complimentary approaches provide a foundation on which to build dynamic adaptive workflow processes.

Introduction

The need for workflows to change and adapt results from a number of causes. Exceptions can result from such sources as inconsistent data [4], divergence of tasks from the underlying workflow model [2], unexpected contingencies [13], and unmodeled changes in the environment [15]. Efforts to evolve, expand, and optimize the workflow process may also be sources of change that must be accommodated by the workflow system [1][11]. Traditional approaches have utilized inflexible control policies that make reactive control and graceful exception handling difficult, if not impossible, tasks.

Our research in development of the Endeavors dynamic web-based workflow system [9][14] has suggested a number of key requirements that should be addressed to support dynamic adaptive workflow. *Contingency management and hand-off* provide mechanisms for dealing with and recovering from expected and unexpected divergence from the intended process. *Partial execution* supports creating and executing processes and process fragments "on-the-fly" as they are needed, rather than requiring the entire process be rigidly specified ahead of time. *Dynamic behaviors*, in terms of both execution model and object behaviors provide flexibility to modify workflow paths and executed behaviors at run-time independent of object data. *Reflexivity* allows a workflow component to programmatically examine, analyze, create, and manipulate its own process and data as part of automatable tasks within the executing workflow [3][5]. Finally, *evolution and optimization* through measurement, tracking, and reuse of generalizable process fragments over the long term can improve the ability of the workflow to adapt to new applications and uses. We discuss each of these in depth in following sections.

Traditional Approaches

Traditional work solutions arise from a number of different areas of research. They possess overlapping constraints which limit their support for dynamic adaptation to change as well as the ease of their integration and adaptation to existing work environments. Even though the trend is toward convergence [6] we address issues of individual approaches here.

Process Technology

Process technologies typically assume closed world data consistency [12]. This is usually enforced by limiting the actions of the user or agent to those only pre-specified by the process designer. In a dynamic, evolving real world environment, sometimes it is necessary to jump out of the process or out of the system to adhere to the specified process model. Process models that are over-specified are difficult to follow because their requirements for completion are too rigid. Process technologies that support evolution as a mechanism to address this problem require that changes are monotonic, i.e. each successive change to the workflow model is additive, incremental and always strongly dependent upon previous models for data integrity and consistency. Consistency sometimes must be violated to meet both functional and non-functional requirements of the process being followed or the product being produced. Coordination between dispersed process participants is sometimes difficult because a uniform representation of activities, artifacts, and resources may differ among people, groups, and organizations. Adding the complication that various participants possess differing skills, different levels of understanding of their obligations, and may require domain and context specific views, misrepresentation and miscommunication is often the result.

Workflow Technology

Traditional workflow technologies have achieved relative success in the workplace through simplification. These systems have been applied to problems of limited scope and scale. Workflow specifications may be ambiguous or contain inflexible semantics. Visual workflow programming languages sometimes lack representations for timing and execution constraints as well as complex relationship specification and management between process objects and people. Lack of relationship management can precipitate poor exception handling. Workflow processes that diverge from the intended series of planned activities or require some midstream changes may result in exceptions. An exception can be something as simple as a missing resource or input; it can also be something more serious such as a design failure requiring significant amount of rework or alternate procedures. Workflow technologies typically lack the infrastructure to query status and change values of their own processes, correct midstream deviations, or locate and reserve needed resources and artifacts. Often exceptions, whether requiring minor automated intervention or human participation to correct, require the reset and restart of a workflow process. Generally, workflow processes once they are deployed into their execution context are not changed. Significant changes may be difficult to accomplish in this environment as the workflow process model, the execution infrastructure, and the process presentation are likely to be tightly coupled.

Groupware and CSCW Technology

Groupware and CSCW are broad labels that describe a gamut of synchronous and asynchronous technologies including email, shared whiteboards, meeting schedulers, collaborative desktops, video conferencing, and other shared electronic media. While these technologies are useful for overcoming communication problems over time and collaboration problems over distance, they lack the guidance and automation mechanisms for performing structured tasks. While using these tools to accomplish unstructured tasks mimics some real world work environments, they do not guarantee consistency of results, maintain a standard of practice and procedure, nor lend themselves to optimization, improvement, and training. There is no explicit work model other than ad hoc utilization of the components at hand. There is no measurable status for determining completion, progress of a task, or even mechanisms for describing what work needs to be done other than capturing communication and collaboration relationships between participants. Adding a work model and management capabilities to a groupware or CSCW technology often leads to additional work on the part of the participant with no direct benefits such as guidance or automation. Synchronization between the proposed work and the actual work is often done by hand. This overhead of model maintenance may lead to the demotivation of the participants which is crucial to the success of an activity.

Approaches for Adaptive Workflow

Contingency Management and Hand-off

Even the best planned workflows will likely encounter exceptions resulting from contingent events during execution. Managing contingencies involves handling the exception and rejoining the "normal" workflow so that it may proceed correctly. While some contingencies can be predicted and planned for, in a complex process it is impossible to plan for all possibilities.

A good recovery mechanism should permit execution of a disrupted workflow to resume, start at an arbitrary midpoint, or rollback to a previous point in the process using computer or human execution of reset, restart, undo, complete, abort, recover, ignore, or jump operations on the process interpreter [8]. Hand-off and recovery of artifacts and reassignment of activities and resources should be localized if permissible. Flexible exception handling that provides scoping mechanisms is a key element in minimizing the detrimental effects of unplanned contingencies. Localized exceptions should not halt the progress of the entire workflow but rather be propagated up hierarchically until the problem can be resolved or an appropriate workaround, such as an alternate path to a solution, can be specified. Alternatively, a "show-stopping" exception, one with global scope, should be broadcast to the appropriate participants and agents to minimize the amount of misdirected work or focus work priorities in the face of new constraints.

To facilitate the workarounds required by contingencies in workflow executions, hand-off of process objects, workflow specifications, and execution context needs to be easily accomplished across networks of dispersed participants. Clear communication channels, unambiguous work and activity model representations, and a common understanding of what is being handed off as well as what expectations are being placed upon the recipient are all requirements for successfully transferring work assignments. This sort of hand-off normally occurs during regular workflow execution, but when an exception occurs, roles and assignments may become ambiguous. Mismatch of expectations may occur, running the risk that important tasks may get lost in the reshuffling of work assignments. Simple sharing of process objects across sites may not be sufficient to guarantee the successful hand-off and resumption of a process due to differences in local work context.

Handshaking may be required to ensure consistency during hand-off. Online, confirmation of items received can be accomplished through electronic dockets or invoices to ensure the existence of items and digital signatures may be used to ensure quality and accuracy. Off-line, a handshake is still a handshake between people, but an online record should reflect the agreement and the social obligations of all participants. Because hand-offs are likely to require changes in execution context, it is important that the workflow processes and objects be designed to adapt to the new environment or gracefully handle contingencies.

Partial Execution

For some projects, even the principle workflow path cannot be completely specified prior to the start of execution. This occurs when downstream details are dependent on upstream results not available when the execution begins. Processes are dynamically composed as execution progresses. In this approach, sometimes referred to as "just in time" execution, the definition of the workflow is not created until needed. Examples of this sort of application include bug-tracking/resolution and experimental or exploratory workflow solutions.

Partial execution supports dynamic composition by allowing the execution of fragments of an incomplete process. Partial execution is analogous to a cat's cradle, a child's game in which an intricately looped string is transferred from the hands of one player to the next, resulting in a succession of different loop patterns. It should be possible to pick up the execution of a process and continue it from any point specified including the dynamic reorganization of local relationships and constraints to fit the new work context. Partial execution supports multiple iterations of a process fragment or multiple alternate iterations of the same process fragment changing order, priority, focus, or completion criteria of the fragment. Support for resolving and integrating processes via pipe-and-filtering, re-stringing, rework, and amalgamation of diverse processes which include possibly competing priorities should be included into the workflow execution infrastructure. This may include temporal and spatial context awareness in addition to the possible execution space and control, coordination, and collaboration policies. For individuals or groups, this may include several partial executions, repeated until the artifact is good enough to post or hand-off to the next participant. While partial execution techniques may create ambiguity and diminish the ability to do global optimizations across all activities before execution, work specifications are generated on-the-fly and on-demand allowing many local optimizations based on discriminates available at execution time.

Dynamic Behaviors

In addition to being able to compose workflows as they are needed, dynamism supports is also important with respect to existing processes. While introduction of dynamic change to a process and its representation may require additional infrastructure to recognize and enforce consistency, limit access to change mechanisms as appropriate, or correct problems that impact other parts of the process, the ability to dynamically evolve in conjunction with the work environment, culture, and context is important to keeping the online description of work consistent with the actual work being performed. Long running, distributed processes involving multiple stakeholders will encounter a multitude of situations demanding change, escalation, and reorganization.

Strong support for dynamic change allows these issues to be addressed with minimal impact to the ongoing, executing workflow process. The ability to dynamically modify a process definition, the set of views into the process, or the execution model, at the time it is in progress, to better fit changing requirements, availability of resources, and the applicability to the current work context is crucial for we described above), but also to workflows whose purpose is to create another process tailored to a specific purpose.

Evolution and Optimization

Evolution of process objects and workflows occurs over time as a result of changing tasks, priorities, responsibilities, and even people. Optimization occurs when the improvement of a previous work model results in a better way of doing things by adding, removing, or redefining process activities and their constraints. As a workflow process is repeatedly executed small changes may be made each time through. Eventually, a process tends to converge on a common practice and become institutionalized.

Unfortunately, a common practice and a best practice may not be the same thing. In order to determine this, metrics must be kept to evaluate one execution from another. This evaluation is subjective because the criteria underlying the metrics changes the same way the workflow does. Successful workflows, like successful software, will be applied in new situations that may have been unanticipated by the original creator, and unsuccessful workflows will be abandoned or changed. It is important in the workflow infrastructure to allow the change to occur both before and after deployment of the workflow, by both technical and non-technical participants. Some optimizations may even be performed by the system itself through agents, validation tools, or optimizers.

To better support process evolution and optimization, process fragments should be easily reusable, divisible, understandable, and capable of being evaluated against some measurable criteria with respect to expected execution or anticipated behavior. A fragment that is successful in one context is likely to be applicable to another similar context. For instance, in a software testing process, the tester may require that the same outcome be reached over repeated executions. Different outcomes imply different qualities of the software. Similarly, a process can be used to develop the skill of a particular student in a training domain where the end-user's path through the process is dependent upon their skill. The inculcation results in the goal of visiting every activity or series of activities through repeated executions and increased skills. The process may change based on feedback, usage, level of interaction, and number of times executed. As with all changing components, change management techniques such as version control and transactions should be integrated with the system.

Customized agents and event monitoring infrastructure are useful for gauging the effectiveness of these workflow components. Further, when a new process is introduced to a work environment and culture, there is often some pushback to its adoption. It is important for the process to be able to adapt in response to this pushback. In addition, process discovery tools are useful for comparing and validating the workflow model with the actual work being accomplished. Wide divergences can indicate technology mismatches or inapplicability and thus the need for evolution and optimization of the process.

Conclusion

Process workflow systems are often called upon to adapt to a changing environment. Exceptions arising from such causes as inconsistencies with the actual process or unexpected occurrences often drive the need to adapt. Alternatively, the need to evolve or optimize the process may drive the changes in the process model. Traditional approaches to handling dynamism in workflow have generally fallen short. Some attempts have proven too rigid, not easily tolerating dynamic change once a process starts executing. Others lack the structure to provide a coherent process model, identify exceptions, guide responses, or manage evolution of processes over time.

We have identified a number of elements from our research as important to supporting dynamic adaptive workflow execution: contingency management and hand-off, partial execution, dynamic behaviors, reflexivity, and long-term evolution and optimization. Taken together, these complimentary emphases provide a foundation on which to build dynamic adaptive workflow processes. Our work with [7] has suggested these are important elements in future workflow systems. We have incorporated many of these approaches into work on the Endeavors workflow system at the University of California, Irvine.

References

1. Abbott, K. and Sarin, S., "Experiences with Workflow Management: Issues for the Next Generation" Proceedings of the Conference on CSCW, Chapel Hill, NC, pp. 113-120, 1994.

2. Bolcer, G., Flexible and Customizable Workflow Execution on the WWW, PhD Thesis, University of California, Irvine. September, 1998.

3. Bolcer, G. and Taylor, R., "Endeavors: A Process System Integration Infrastructure", 4th International Conference on Software Process, Brighton, UK, December, 1996.

4. Cugola, G., Di Nitto, E., Fuggetta, A., and Ghezzi, C. "A Framework for Formalizing Inconsistencies and Deviations in Human-Centered Systems" ACM Transactions on Software Engineering and Methodology (TOSEM), 5(3), ;pp.191-230, 1996.

5. Dourish, P., "Developing a Reflective Model of Collaborative Systems", ACM Transactions on Computer-Human Interactions, vo.2, no.1, pp.40-63, March, 1995.

6. Ellis, C. and Nutt, G., "Workflow: The Process Spectrum, NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions", Athens, Georgia, pp. 140-145, May, 1996.

7. Fielding, R. et al. "Web-Based Development of Complex Information Products", Communications of the ACM, vol. 41, no. 8, August, 1998.

8. Kaiser, G. et al. "WWW-based Collaboration Environments with Distributed Tool Services", World Wide Web Journal, Baltzer Science Publishers, January, 1998.

9. Kammer, P. et al., "Supporting Distributed Workflow Using HTTP", 5th International Conference on Software Process, Chicago, June, 1998.

10. Miller, J. et al. "The Future of Web-based Workflows" LDIS Department of Computer Science, University of Georgia, Athens, Research Directions in Process Technology Workshop, Nancy, France, July, 1997.

11. Nutt, G., "The Evolutions Toward Flexible Workflow Systems" Distributed Systems Engineering, vol. 3, no. 4, pp. 276-294, December, 1996.

12. Osterweil, L., "Software Processes are Software Too, Revisited", In Proceedings of the International Conference on Software Engineering, Boston, MA., pp. 540-548, May, 1998.

13. Saastamoinen, H. et al., "Survey on Exceptions in Office Information Systems" Technical Report CU-CS-712-95, Department of Computer Science, University of Colorado, Boulder, 1994.

14. Taylor, R. "Dynamic, Invisible, and on the Web", University of California, Irvine, Research Direction in Process Technology Workshop, Nancy, France, July, 1997.

15. Tolone, W., "Introspect: a Meta-Level Specification Framework for Dynamic Evolvable Collaboration Support", PhD Thesis. University of Illinois at Urbana-Champaign, 1996.